REMARKS/ARGUMENTS

Claims 1-6 and 14 are pending in the present application. Claims 1-6 and 14 were amended, and claims 7-13 and 15-20 were canceled. No claims were added. Support for the claim amendments can be found, for example, in the specification on page 9, line 40 to page 10, line19, and in Figure 4. Reconsideration is respectfully requested in view of the above amendments and the following comments.

In this Amendment, Applicants have amended claims 1-6 and 14 and canceled claims 7-13 and 15-20 from further consideration in this application. Applicants are not conceding that the subject matter encompassed by claims 1-20 prior to this Amendment is not patentable. Claims 1-6 and 14 were amended and claims 7-13 and 15-20 were canceled in this Amendment solely to facilitate expeditious prosecution of the application. Applicants respectfully reserve the right to pursue additional claims, including the subject matter encompassed by claims 1-20 as presented prior to this Amendment and additional claims in one or more continuing applications.

I. 35 U.S.C. § 101

The Examiner has rejected claims 13 and 16-20 under 35 U.S.C. § 101 as being directed towards non-statutory subject matter. This rejection is respectfully traversed.

In rejecting the claims, the Examiner states:

> Claim 13, and 16-20 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. The "computer program product" of these claims is interpreted to be directed to Compute Software *per se*. Computer Software *per se* is considered functional descriptive material and therefore non-statutory when not claimed in combination with sufficient structure to render the claim statutory. Please see MPEP §2106.

Office Action dated February 20, 2008, page 2.

By the present amendment, claims 13 and 16-20 have been canceled. Therefore, the rejection of claims 13 and 16-20 under 35 U.S.C. § 101 is now moot.

II. 35 U.S.C. § 102, Anticipation

The Examiner has rejected claims 1-20 under 35 U.S.C. § 102 as being anticipated by Fresko et al., U.S. Patent Number 5,966,702 (hereinafter referred to as "Fresko"). This rejection is respectfully traversed.

In rejecting the claims, the Examiner states:

> Regarding Claim 1, Fresko teaches: A data processing method for creating an executable file by combining a plurality of run units, the method comprising the steps of: reading a first run unit to be added to the executable file **(Column 9, Lines 15-17, "The method begins in step 400 with a set of arbitrary class files "S" (typically part of one application). In step 401, the pre-processor reads and parses each class in "S."")**; locating a first data entity set to a first string value in the first run unit **(Column 9, Lines 17-21, "In step 402, the pre-processor examines the constant pool tables of each class to determine the set of class file constants (such as strings and numerics, as well as others specific to the class file format) that can be shared between classes in "S."")**; matching the first data entity with a second data entity set to a second string value, the second data entity being from a second run unit previously added to the executable file **(Column 9, Lines 21-23, "A shared constant pool table is created in step 403, with all duplicate constants determined from step 402.")**; and adding the first run unit to the executable file but without the first data entity **(Column 9, Lines 23-35, "In step 404, the pre-processor removes the duplicate, shared constants from the individual constant pool tables of each class.")**

Office Action dated February 20, 2008, pp. 2-3.

Claim 1, as amended herein, is as follows:

> 1. A data processing method for creating an executable file by combining a plurality of run units, the method comprising:
>
> responsive to a determination that a first run unit to be added to the executable file comprises a first data entity set to a first value indicating that the first data entity is required to appear only once in the executable file, determining whether the first data entity matches a second data entity set to a second value and included in a second run unit, wherein the second run unit comprises a run unit that was previously added to the executable file;
>
> responsive to a determination that the first data entity matches the second data entity, adding the first run unit to the executable file without the first data entity; and
>
> responsive to a determination that the first data entity does not match the second data entity, adding the first run unit to the executable file with the first data entity.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). In this case, each and every feature of the presently claimed invention is not identically shown in Fresko, arranged as they are in the claims, and,

accordingly, Fresko does not anticipate the claims.

Fresko is generally directed to a mechanism for preprocessing and packaging class files so as to remove duplicate information elements from a set of class files (see Abstract in Fresco). The Examiner cites to various recitations in column 9, lines 15-35 of Fresko as disclosing claim 1. Column 9, lines 15-35 of Fresko is reproduced below for the convenience of the Examiner:

> The method begins in step **400** with a set of arbitrary class files "S" (typically part of one application). In step **401**, the pre-processor reads and parses each class in "S." In step **402**, the pre-processor examines the constant pool tables of each class to determine the set of class file constants (such as strings and numerics, as well as others specific to the class file format) that can be shared between classes in "S." A shared constant pool table is created in step **403**, with all duplicate constants determined from step **402**. In step **404**, the pre-processor removes the duplicate, shared constants from the individual constant pool tables of each class.
>
> In step **405**, the pre-processor computes the in-core memory requirements of each class in "S," as would normally be determined by the class loader for the given virtual machine. This is the amount of memory the virtual machine would allocate for each class, if it were to load each class separately. After considering all classes in "S" and the additional memory requirement for the shared constant pool table, the total memory requirement for loading "S" is computed in step **406**."

As described in the above-reproduced portion, Fresko provides a pre-processor that parses a set of arbitrary class files and examines constant pool tables of each class to determine a set of class file constants that can be shared between classes, and a shared constant pool table is created with all duplicate constants that were determined.

Fresko does not, however, disclose or suggest "responsive to a determination that a first run unit to be added to the executable file comprises a first data entity set to a first value indicating that the first data entity is required to appear only once in the executable file, determining whether the first data entity matches a second data entity set to a second value and included in a second run unit, wherein the second run unit comprises a run unit that was previously added to the executable file" as recited in amended claim 1. Fresco does not compare a data entity included in a run unit to be added to an executable file with a data entity included in a run unit that was previously added to the executable file, and does not make a determination whether a first data entity included in a first run unit to be added to an executable file matches a second data entity included in a second run unit that was previously added to the executable file.

Yet further, because Fresko does not disclose or suggest "determining whether the first data entity matches a second data entity set to a second value and included in a second run unit, wherein the second run unit comprises a run unit that was previously added to the executable file" as recited in claim 1, the reference also does not disclose or suggest making such a determination "responsive to a determination that a first run unit to be added to the executable file comprises a first data entity set to a first value

indicating that the first data entity is required to appear only once in the executable file" as is additionally recited in claim 1.

Fresko, accordingly, does not disclose or suggest "responsive to a determination that a first run unit to be added to the executable file comprises a first data entity set to a first value indicating that the first data entity is required to appear only once in the executable file, determining whether the first data entity matches a second data entity set to a second value and included in a second run unit, wherein the second run unit comprises a run unit that was previously added to the executable file" as recited in amended claim 1, and does not anticipate claim 1 for this reason.

Fresko also does not disclose or suggest "responsive to a determination that the first data entity matches the second data entity, adding the first run unit to the executable file without the first data entity", and "responsive to a determination that the first data entity does not match the second data entity, adding the first run unit to the executable file with the first data entity" as now recited in claim 1. Instead, as described in the above-reproduced portion of Fresko, a shared constant pool table is created with all duplicate constants, and the duplicate, shared constants are removed from individual constant pool tables of each class. Fresko does not disclose or suggest adding a run unit to an executable file without or with a data entity depending on whether the data entity matches a data entity that was included in a run unit previously added to the executable file.

Fresko, accordingly, also does not disclose or suggest "responsive to a determination that the first data entity matches the second data entity, adding the first run unit to the executable file without the first data entity", and "responsive to a determination that the first data entity does not match the second data entity, adding the first run unit to the executable file with the first data entity", and does not anticipate claim 1for this reason as well.

Claim 1 as amended herein, accordingly, is not anticipated by Fresko and patentably distinguishes over Fresko in its present form.

Claims 2-6 and 14 depend from and further restrict claim 1 and are also not anticipated by Fresko, at least by virtue of their dependency. In addition, the claims recite further features that are not disclosed or suggested by Fresko. For example, claim 3 recites that the first data entity matches the second data entity if the second value partially matches the first value. As noted by the Examiner, Fresko discloses only that "[a] shared constant pool table is created in step 403, with all duplicate constants determined from step 402." Claim 3, accordingly, is not anticipated by Fresko in its own right as well as by virtue of its dependency.

Claim 4 depends from claim 3 and is as follows:

> 4. A method of claim 3 further comprising:
> determining whether the first data entity matches a third data entity included in a third run unit to be added to the executable file, wherein the third data entity is set to a third value indicating that the third data entity is required to appear only once in the executable file, and wherein the first data entity matches the third data entity if the third value contains the first value;
> responsive to a determination that the first data entity matches the third data entity, removing the first data entity from the executable file; and
> adding the third data entity to the executable file.

In rejecting claim 4, the Examiner states:

> Regarding Claim 4, 10 and 18 Fresko further teaches: further comprising the steps: reading a third run unit to be added to the executable file, wherein the third run unit contains a third data entity of a third string value (**Column 9, Lines 21-23, "A shared constant pool table is created in step 403, with all duplicate constants determined from step 402."**); matching the first data entity with the third data entity wherein a match is found if the third string value contains the first string value (**Column 9, Lines 21-23, "A shared constant pool table is created in step 403, with all duplicate constants determined from step 402."**); removing the first data entity from the executable file (**Column 9, Lines 23-35, "In step 404, the pre-processor removes the duplicate, shared constants from the individual constant pool tables of each class."**); and adding the third data entity to the executable file (**Column 9, Lines 23-35, "In step 404, the pre-processor removes the duplicate, shared constants from the individual constant pool tables of each class."**) .

Office Action dated February 20, 2008, pp. 5-6.

Fresko discloses only that when the pre-processor creates a shared constant pool table, duplicate shared constants are removed from individual constant pool tables. The reference does not disclose or suggest removing a first data entity from an executable file; and adding a third data entity to the executable file "responsive to a determination that the first data entity matches the third data entity" as recited in claim 4. Claim 4, accordingly, is also not anticipated by Fresko in its own right as well as by virtue of its dependency.

For at least all the above reasons, claims 1-6 and 14 are not anticipated by Fresko and patentably distinguish over Fresko in their present form. Claims 7-13 and 15-20 have been canceled and the rejection with respect to those claims is now moot. Therefore, the rejection of claims 1-20 under 35 U.S.C. § 102 has been overcome.

III. Conclusion

It is respectfully urged that the subject application is patentable over the cited reference and is now in condition for allowance. It is, accordingly, respectfully requested that the Examiner so find and issue a Notice of Allowance in due course.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: May 20, 2008

Respectfully submitted,

/Gerald H. Glanzman/

Gerald H. Glanzman
Reg. No. 25,035
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants